Algorithmica (2008) 51: 239–263
DOI 10.1007/s00453-007-9111-9

Generating Cut Conjunctions in Graphs and Related Problems

Leonid Khachiyan · Endre Boros · Konrad Borys · Khaled Elbassioni · Vladimir Gurvich · Kazuhisa Makino

Received: 9 February 2006 / Accepted: 6 November 2006 / Published online: 27 October 2007

Abstract Let $G = (V, E)$ be an undirected graph, and let $B \subseteq V \times V$ be a collection of vertex pairs. We give an incremental polynomial time algorithm to generate all minimal edge sets $X \subseteq E$ such that every pair $(s, t) \in B$ of vertices is disconnected in $(V, E \setminus X)$, generalizing well-known efficient algorithms for generating all minimal s-t cuts, for a given pair s, t of vertices. We also present an incremental polynomial time algorithm for generating all minimal subsets $X \subseteq E$ such that no $(s, t) \in B$ is

This research was partially supported by the National Science Foundation (Grant IIS-0118635), by DIMACS, the National Science Foundation's Center for Discrete Mathematics and Theoretical Computer Science, and by the Scientific Grant-in-Aid from the Ministry of Education, Science, Sports and Culture of Japan.
Our friend and colleague, Leonid Khachiyan passed away with tragic suddenness while we were preparing this manuscript.

L. Khachiyan
Department of Computer Science, Rutgers University, 640 Bartholomew Road, Piscataway, NJ 08854-8003, USA

E. Boros (✉) · K. Borys · V. Gurvich
RUTCOR, Rutgers University, 640 Bartholomew Road, Piscataway, NJ 08854-8003, USA
e-mail: boros@rutcor.rutgers.edu

K. Borys
e-mail: kborys@rutcor.rutgers.edu

V. Gurvich
e-mail: gurvich@rutcor.rutgers.edu

K. Makino
Department of Mathematical Informatics, Graduate School of Information and Technology, University of Tokyo, Tokyo, 113-8656, Japan
e-mail: makino@mist.i.u-tokyo.ac.jp

K. Elbassioni
Max-Planck-Institut für Informatik, Saarbrücken, Germany
e-mail: elbassio@mpi-sb.mpg.de

a bridge in $(V, X \cup B)$. Both above problems are special cases of a more general problem that we call generating cut conjunctions for matroids: given a matroid M on ground set $S = E \cup B$, generate all minimal subsets $X \subseteq E$ such that no element $b \in B$ is spanned by $E \setminus X$. Unlike the above special cases, corresponding to the cycle and cocycle matroids of the graph $(V, E \cup B)$, the more general problem of generating cut conjunctions for vectorial matroids turns out to be NP-hard.

Keywords Cut conjunction · Cut generation · Graph · Matroid · Multicut

1 Introduction

Given a graph $G = (V, E)$ and two vertices $s, t \in V$, the *two-terminal cut generation problem* calls for listing all minimal subsets of edges whose removal disconnects s and t. This problem is known to be solvable in $O(Nm + m + n)$ time and $O(n + m)$ space [16], where n and m are the numbers of vertices and edges in the input graph, and N is the total number of cuts. In this paper, we study the following natural extension of this problem:

Generating Cut Conjunctions in Graphs
Input: An undirected graph $G = (V, E)$, and a collection $B = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of k pairs of vertices $s_i, t_i \in V$
Output: The list of all minimal edge sets $X \subseteq E$ such that for all $i = 1, \ldots, k$, vertices s_i and t_i are disconnected in $G' = (V, E \setminus X)$

Note that for $i \neq j$, s_i and s_j, or s_i and t_j, or t_i and t_j may coincide. We call a minimal edge set $X \subseteq E$ for which all pairs of vertices $(s_i, t_i) \in B$ are disconnected in the subgraph $G' = (V, E \setminus X)$, a *minimal B-cut*, or simply a *cut conjunction* if B is clear from the context.

Let $\mathcal{F}$ denote the family of all minimal B-cuts. Observe that each edge set $X \in \mathcal{F}$ must indeed be the union of some minimal s_i–t_i cuts for $i = 1, \ldots, k$, justifying the name "cut conjunction". Note also that not all unions of minimal s_i–t_i cuts for $i = 1, \ldots, k$ are minimal B-cuts. Figure 1 depicts a graph with the number of minimal s_k–t_k cuts not bounded polynomially in $|V|$ and $|\mathcal{F}|$, showing that the generation of cut conjunctions cannot be efficiently reduced to two-terminal cut generation.

Without any loss of generality we can assume for each $i = 1, \ldots, k$ that (i) the pair of vertices s_i and t_i are in the same connected component of G, since otherwise the pair (s_i, t_i) could simply be deleted from B without changing the problem, and (ii) vertices s_i and t_i are not adjacent in G, since otherwise the edge $s_i t_i$ would belong to all cut conjunctions.

When B is the collection of all pairs of distinct vertices drawn from some vertex set $V' \subseteq V$, minimal B-cuts are known as *multiway cuts*, see e.g., [7, 17]. The optimization problem of finding a minimum weight multiway cut is known to be NP-hard for $|V'| \geq 3$ [4]. On the other hand, the generation of multiway cuts is a special case of the generation of cut conjunctions in graphs, which turns out to be tractable, in the sense it is defined at the end of this section.

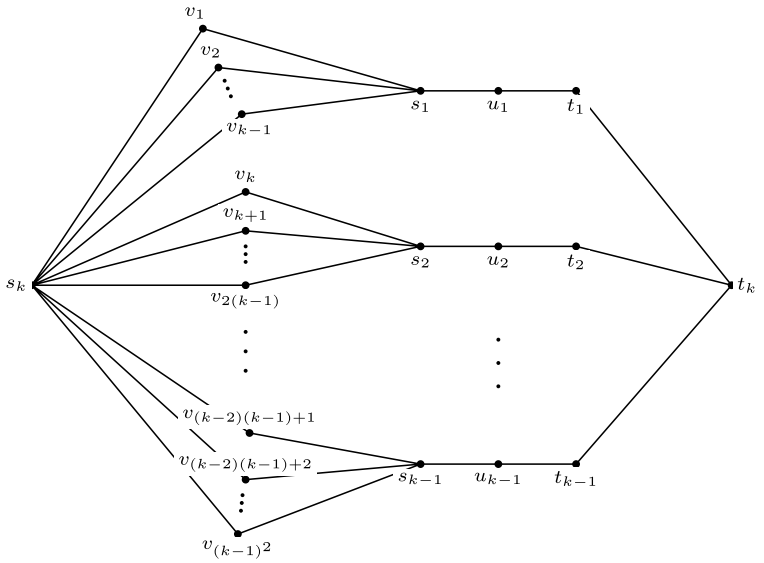

Fig. 1 Minimal B-cuts contain exactly one edge of each pair $s_i u_i$ and $u_i t_i$, for $i = 1, \ldots, k-1$, thus we have $|\mathcal{F}| = 2^{k-1}$. On the other hand, the number of minimal s_k–t_k cuts is more than $2^{(k-1)^2}$, so it is not polynomial in $|V| = k^2 + k$ and $|\mathcal{F}|$

It will be convenient to consider generating cut conjunctions of graphs in the context of the more general problem of generating cut conjunctions of (vectorial) matroids. In what follows we assume familiarity with matroid theory (see e.g., [10, 18] for a thorough introduction).

Generating Cut Conjunctions in Matroids
Input: A matroid M on ground set S and a set $B \subseteq S$
Output: The list of all maximal sets $X \subseteq A \stackrel{\text{def}}{=} S \setminus B$ that span no element of B

When M is the cycle matroid of a graph $G = (V, E \cup B)$, where $E \cap B = \emptyset$, we can let $S = E \cup B$, and then by definition, an edge set $Y \subseteq A = E$ spans $b = (s_i, t_i) \in B$ if and only if Y contains an s_i–t_i path. This means that a maximal edge set $Y \subseteq E$ spans no edge $b \in B$ in the matroid M if and only if $X = E \setminus Y$ is a minimal B-cut in the graph (V, E). Thus, the problem of generating cut conjunctions in graphs is a special case of the problem of generating cut conjunctions in matroids.

Let $r : S \to \mathbb{Z}\_+$ be the rank function of a matroid M on S. The *dual* matroid M^* on S is defined by the rank function $r^*(X) = r(S \setminus X) + |X| - r(S)$, see e.g., [18]. In particular, $Y \subseteq A = S \setminus B$ spans $b \in B$ in M^* if and only if $r^*(Y \cup \{b\}) = r^*(Y)$, which is equivalent to $r(X \cup B) = r(X \cup (B \setminus \{b\})) + 1$. This means that generating cut conjunctions for the dual matroid M^* is equivalent to the following generation problem:

Generating Cut Conjunctions in the Dual Matroid
Input: A matroid M on ground set S and a subset $B \subseteq S$
Output: The list of all minimal sets $X \subseteq A \stackrel{\text{def}}{=} S \setminus B$ such that each element $b \in B$ is spanned by $X \cup (B \setminus \{b\})$

In particular, when M is the cycle matroid of a graph $G = (V, E)$ (and consequently, M^* is the cocycle matroid of G), the dual formulation can be restated as follows:

Generating Bridge-Avoiding Extensions in Graphs
Input: An undirected graph $G = (V, E)$ and a collection of edges $B \subseteq E$
Output: The list of all minimal edge sets $X \subseteq E \setminus B$ such that no edge $b \in B$ is a bridge in $G' = (V, B \cup X)$

Note that in all of the mentioned generation problems, the output, $\mathcal{F}$, may consist of exponentially many sets, in terms of the input size. Therefore we measure the running time of generation algorithms in both the input and output size. A generation algorithm may output an element of $\mathcal{F}$ any time during its execution. A generation algorithm runs in *incremental polynomial time* if it outputs K elements of $\mathcal{F}$ (or all, if $|\mathcal{F}| < K$) in time polynomial in the input size and K. A generation algorithm runs with *polynomial delay* if it outputs K elements of $\mathcal{F}$ (or all) in time polynomial in the input size and linear in K (see e.g., [8, 9]).

2 Main Results

We show that all of the above generation problems for graphs can be solved in incremental polynomial time. Let $G = (V, E)$ be a graph, $|V| = n$, $|E| = m$, and $k = |B|$.

Theorem 1 *For every K we can generate K (or all, if there are no more than K) cut conjunctions of G in $O(K^2 \log(K) nm^2 + K^2 k(n+m)m^2)$ time.*

Theorem 2 *For every K we can generate K (or all, if there are no more than K) bridge-avoiding extensions of G in $O(K^2 \log(K) m^2 + K^2 m^2 (n+m))$ time.*

In contrast, we recall that generating cut conjunctions in matroids is an NP-hard problem:

Proposition 3 ([2]) *Let M be a vectorial matroid defined by a collection S of n-dimensional vectors over a field of characteristic zero or of large enough characteristic (at least $8n$), let B be a subset of S and let $\mathcal{F}$ be the family of all maximal subsets of $A \stackrel{\text{def}}{=} S \setminus B$ that span no vector $b \in B$. Given a subfamily $\mathcal{X} \subseteq \mathcal{F}$, it is NP-hard to decide if $\mathcal{X} \neq \mathcal{F}$.*

In addition to indicating that generating cut conjunctions in vectorial matroids is NP-hard, the above result also implies that the dual formulation is NP-hard, too. This follows from the fact that the dual M^* of an explicitly given vectorial matroid M over a field $\mathbf{F}$ is again a vectorial matroid over the same field. Moreover, an explicit representation for M^* can be obtained efficiently from the given representation of M (see [12]).

As stated in Proposition 3, our NP-hardness result for generating cut conjunctions in vectorial matroids is valid over sufficiently large fields. In particular, the complexity of generating cut conjunctions in *binary* matroids remains open. We can only show that this problem is tractable for $|B| = 2$:

Proposition 4 *Let M be a binary matroid on ground set S and let $B = \{b_1, b_2\} \subseteq S$. All maximal subsets X of $A \stackrel{\text{def}}{=} S \setminus B$ that span neither b_1 nor b_2 can be generated in incremental polynomial time.*

Finally, it is worth mentioning that generating cut conjunctions in binary matroids includes, as a special case, the well-known *hypergraph dualization* problem [5, 6]:

Generating Minimal Transversals of a Hypergraph
Input: A hypergraph $\mathcal{H}$.
Output: The list of all minimal transversals (equivalently, maximal independent sets) of $\mathcal{H}$.

To see this inclusion, let us consider the following construction. Let B be the $n \times |\mathcal{H}|$ binary matrix whose columns are the characteristic vectors of the hyperedges of $\mathcal{H}$, and let I be the $n \times n$ identity matrix. Letting $M = [I, B]$ and denoting by A the columns set of I, we can readily identify each maximal subset of A that spans no columns of B with a maximal independent vertex set of $\mathcal{H}$. This shows that generating cut conjunctions for a binary matroid is at least as hard as generating all maximal independent sets for a hypergraph. The theoretically fastest currently available algorithm for hypergraph dualization generates all maximal independent sets in incremental quasi-polynomial time [6].

The remainder of the paper is organized as follows: In the next section we describe a general approach for generation problems. We prove Theorems 1 and 2 respectively in Sects. 4 and 5. In Sect. 6 we prove Proposition 4, and for completeness we include the proof of Proposition 3 in the Appendix (an alternative proof can be found in [2]).

3 The $X - e + Y$ Method

In this section we present a technique which is a variant of the so called *supergraph approach* that has been used in the literature for instance, to generate all minimal feedback vertex and arc sets [13], minimal s–t cuts [16], minimal spanning trees [14], and minimal blockers of perfect matchings in bipartite graphs [3]. To explain the method briefly, a supergraph is a strongly connected directed graph $\mathcal{G}$ whose vertices

are the objects that we would like to generate. We can arrive to such a directed graph by appropriately defining the out-neighborhood of each object. Once we have an efficient way of generating such a neighborhood, and they define a strongly connected directed graph, then we can generate all objects simply by traversing $\mathcal{G}$.

In this section we present a variant of this general approach, which we call the $X - e + Y$ method. To formulate this method, let us consider a general framework for generation problems. Let E be a finite set and $\pi_E : 2^E \to \{0, 1\}$ be a monotone Boolean function, i.e., one for which $X \subseteq Y$ implies $\pi_E(X) \leq \pi_E(Y)$. We shall assume that $\pi_E(\emptyset) = 0$ and $\pi_E(E) = 1$. Let us then define a family $\mathcal{F}$ as follows

$$\mathcal{F} = \{X \mid X \subseteq E \text{ is a minimal set satisfying } \pi_E(X) = 1\},$$

and consider the goal of generating all sets belonging to $\mathcal{F}$.

This goal can be achieved by the following $X - e + Y$ method. To simplify notation, we write in the sequel $X \cup e$ and $X \setminus e$ instead of $X \cup \{e\}$ and $X \setminus \{e\}$, respectively.

First we fix an arbitrary linear order $\prec$ on elements of E and define a projection $\Pi : \{X \subseteq E \mid \pi_E(X) = 1\} \to \mathcal{F}$ by $\Pi(X) = X \setminus Z$, where Z is the lexicographically first subset of X, with respect to $\prec$, such that $\pi_E(X \setminus Z) = 1$ and $\pi_E(X \setminus (Z \cup e)) = 0$ for every $e \in X \setminus Z$. We can compute $\Pi(X)$ by deleting one by one in their $\prec$ order the elements of X, whose removal does not change the value of $\pi_E(X)$. This requires evaluating the function π_E exactly $|X|$ times.

We next introduce a directed graph $\mathcal{G} = (\mathcal{F}, \mathcal{E})$ on vertex set $\mathcal{F}$. We define the neighborhood $N(X)$ of a vertex $X \in \mathcal{F}$ as follows

$$N(X) = \{\Pi((X \setminus e) \cup Y) \mid e \in X, Y \in \mathcal{Y}_{X,e}\},$$

where $\mathcal{Y}_{X,e}$ is defined by

$$\mathcal{Y}_{X,e} = \{Y \mid Y \text{ is a minimal subset of } E \setminus X \text{ satisfying } \pi_E((X \setminus e) \cup Y) = 1\}.$$

In other words, for every set $X \in \mathcal{F}$ and for every element $e \in X$ we extend $X \setminus e$ in all possible minimal ways to a set $X' = (X \setminus e) \cup Y$ for which $\pi_E(X') = 1$ (since $X \in \mathcal{F}$, we have $\pi_E(X \setminus e) = 0$), and introduce each time a directed arc from X to $\Pi(X')$. We call the obtained directed graph $\mathcal{G}$ the *supergraph* of our generation problem.

Lemma 5 *For all subsets $X \in \mathcal{F}$, elements $e \in X$ and sets $Y \in \mathcal{Y}_{X,e}$ we have $\Pi((X \setminus e) \cup Y) \setminus (X \setminus e) = Y$.*

Proof By the minimality of Y, we have $\pi_E((X \setminus e) \cup (Y \setminus y)) = 0$ for every $y \in Y$. Thus $\Pi((X \setminus e) \cup Y)$ must contain Y, and by definition, it cannot contain any other elements outside $X \setminus e$. □

Proposition 6 *The supergraph $\mathcal{G} = (\mathcal{F}, \mathcal{E})$ is strongly connected.*

Proof Let $X, X' \in \mathcal{F}$ be two vertices of $\mathcal{G}$. We show by induction on $|X \setminus X'|$ that $\mathcal{G}$ contains a directed path from X to X'. If $X \setminus X' = \emptyset$ then $X \subseteq X'$, but since X' is minimal, $X = X'$ must follow.

Suppose that $|X \setminus X'| > 0$. We show that there is a neighbor X'' of X such that $|X'' \setminus X'|$ is smaller than $|X \setminus X'|$. For this, we choose an arbitrary element $e \in X \setminus X'$. Since $(X \setminus e) \cup X'$ contains X' and $\pi_E(X') = 1$, we have $\pi_E((X \setminus e) \cup X') = 1$ by the monotonicity of π_E. Hence there is a minimal nonempty set $Y \subseteq X' \setminus X$ such that $\pi_E((X \setminus e) \cup Y) = 1$. Now let $X'' = \Pi((X \setminus e) \cup Y)$ be a neighbor of X. By Lemma 5, we have $X'' = (X \setminus (Z \cup e)) \cup Y$. Thus $|X'' \setminus X'| \leq |X \setminus (X' \cup e)| < |X \setminus X'|$. □

Since $\mathcal{G}$ is strongly connected, by performing a breadth-first search in $\mathcal{G}$ we can generate all elements of $\mathcal{F}$. Thus, given a procedure that generates all elements of $\mathcal{Y}_{X,e}$ for every $X \in \mathcal{F}$ and $e \in X$, the procedure *Transversal*($\mathcal{G}$), defined below, generates all elements of $\mathcal{F}$.

Traversal($\mathcal{G}$)

Find the initial vertex $X^0 \leftarrow \Pi(E)$, initialize a queue $\mathcal{Q} = \emptyset$ and a dictionary of output vertices $\mathcal{D} = \emptyset$.
Perform a breadth-first search of $\mathcal{G}$ starting from X^0:

```
1 output X^0 and insert it to Q and to D
2 while Q ≠ ∅ do
3   take the first vertex X out of the queue Q
4   for every e ∈ X do
5     for every Y ∈ Y_{X,e}
6       compute the neighbor X' ← Π((X \ e) ∪ Y)
7       if X' ∉ D then output X' and insert it to Q and to D
```

Lemma 7 *If Y and Y' are distinct elements of $\mathcal{Y}_{X,e}$, then they produce different neighbors of X in $\mathcal{G}$ in line* **6**.

Proof First we observe that for every $Y \in \mathcal{Y}_{X,e}$ we have $\Pi((X \setminus e) \cup Y) = ((X \setminus (Z \cup e)) \cup Y$, where Z is the lexicographically first subset of $X \setminus e$, with respect to $\prec$, such that $\pi_E((X \setminus (Z \cup e)) \cup Y) = 1$ and $\pi_E((X \setminus (Z \cup e \cup f)) \cup Y) = 0$ for every $f \in X \setminus (Z \cup e)$. By the minimality of Y, we have $\pi_E((X \setminus e) \cup (Y \setminus y)) = 0$ for every $y \in Y$. Thus $\Pi((X \setminus e) \cup Y)$ must contain Y. Also note that by minimality of Y, we obtain $X \setminus e$ and Y are disjoint.

Hence for Y and Y', distinct elements of $\mathcal{Y}_{X,e}$, we have $\Pi((X \setminus e) \cup Y) = (X \setminus (Z \cup e)) \cup Y$ and $\Pi((X \setminus e) \cup Y') = (X \setminus (Z' \cup e)) \cup Y'$. Since $\Pi((X \setminus e) \cup Y)$ contains Y, $\Pi((X \setminus e) \cup Y')$ contains Y', $X \setminus e$ and Y are disjoint, $X \setminus e$ and Y' are disjoint and $Y \neq Y'$ we obtain $\Pi((X \setminus e) \cup Y) \neq \Pi((X \setminus e) \cup Y')$. □

Proposition 8 *Assume that there is a procedure that outputs K elements of $\mathcal{Y}_{X,e}$ in time $\phi(K, E)$ and there is an algorithm evaluating π_E in time $\gamma(E)$. Then Traversal($\mathcal{G}$) outputs K elements of $\mathcal{F}$ in time $O(K^2|E|^2\gamma(E) + K^2 \log(K)|E|^2 + K|E|\phi(K, E))$.*

Proof Let $X \in \mathcal{F}$ and $e \in X$. Note that we output a vertex of the supergraph $\mathcal{G}$ every time we insert it to the queue $\mathcal{Q}$ and each vertex of $\mathcal{G}$ is inserted to the queue $\mathcal{Q}$ and removed from $\mathcal{Q}$ only once. Thus to generate K elements we repeat the while loop of lines **2–7** at most K times. As $|X| < |E|$ we repeat the for loop of lines **4–7** at most $|E|$ times. By Lemma 7 we repeat the for loop of lines **5–7** at most K times (otherwise we generate more than K distinct neighbors). Generating K elements of $\mathcal{Y}_{X,e}$ takes $\phi(K, E)$ time.

We repeat lines **6**, **7** at most $K^2|E|$ times. Recall that evaluating *Project* takes $|E|\gamma(E)$ time. We can implement the dictionary $\mathcal{D}$ as a balanced binary search tree. Then the operations FIND and INSERT in $\mathcal{D}$ require at most a logarithmic number of comparisons, where each comparison takes $O(|E|)$ time. This implies that executing lines **6**, **7** a single time takes $O(|E|\gamma(E) + \log(K)|E|)$ time.

Thus the time *Traversal*$(\mathcal{G})$ needs to output K elements is $O(K^2|E|^2\gamma(E)+ K^2\log(K)|E|^2 + K|E|\phi(K, E))$. □

To illustrate the $X - e + Y$ method, let us consider the following problem from [1]:

Path Conjunctions Generation Problem

Input: An undirected graph $G = (V, E)$ and a collection $B = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of k vertex pairs $s_i, t_i \in V$

Output: The list of all minimal edge sets $X \subseteq E$ such that t_i is reachable from s_i in (V, X) for all $i = 1, \ldots, k$

We call such edge set X a *path conjunction*. As shown in [1] path conjunctions can be generated in incremental polynomial time. Here we show that the $X - e + Y$ method provides a simple alternative algorithm. More precisely, for every K we can generate K (or all, if their number is less than K) path conjunctions of a given graph in $O(K^2\log(K)m^2 + K^2m^2k(n + m))$ time, where as before n and m denote the number of vertices and edges of G, respectively.

First, we define a Boolean function π_E as follows: for a subset $X \subseteq E$ let

$$\pi_E(X) = \begin{cases} 1, & \text{every } t_i \text{ is reachable from } s_i \text{ in } (V, X); \\ 0, & \text{otherwise.} \end{cases}$$

Clearly π is monotone and $\mathcal{F} = \{X \mid X \subseteq E \text{ is a minimal set satisfying } \pi_E(X) = 1\}$ is the family of all minimal path conjunctions. We can test if t_i is reachable from s_i applying breadth first search. Thus $\gamma(E) = O(k(n + m))$. We next show that we can generate K (or all) elements of $\mathcal{Y}_{X,e}$ in $\phi(K, E) = O(Km + n + m)$ time.

Let $X \in \mathcal{F}$. We observe that X is a collection of vertex-disjoint trees $\mathcal{T}$ such that for each vertex pair (s_i, t_i) there is a tree containing both s_i and t_i. Removing an edge e from X splits a tree $T \in \mathcal{T}$ containing e into two subtrees T', T''. By the minimality of X there is at least one pair of B with one vertex belonging to T' and the other to T''.

Let G' be the graph obtained from G by contracting each tree of $\mathcal{T} \setminus T$ and T', T'' into a vertex, and let u and v denote the vertices of G' corresponding to T' and T''.

A minimal edge set Y restores that every t_i is reachable from s_i in $(V, (X \setminus e) \cup Y)$ if and only if Y is a path from u to v in G'.

Thus $\mathcal{Y}_{X,e}$ is the family of all u–v paths in G', where G' has at most n vertices and m edges. K paths between two vertices in a graph can be generated via backtracking in $O(Km+n+m)$ time [11]. Consequently, by Proposition 8 *Traversal*($\mathcal{G}$) generates K (or all) path conjunctions in $O(K^2 \log(K)m^2 + K^2m^2k(n+m))$ time.

4 Proof of Theorem 1

In this section we apply the $X - e + Y$ method to the generation of all cut conjunctions.

Given a graph $G = (V, E)$, a collection $B = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of k pairs of vertices $s_i, t_i \in V$, and a subset $X \subseteq E$, we define a Boolean function π as follows: for a subset $X \subseteq E$ let

$$\pi(X) = \begin{cases} 1, & s_i \text{ is disconnected from } t_i \text{ in } (V, E \setminus X) \text{ for all } i = 1, \ldots, k; \\ 0, & \text{otherwise.} \end{cases}$$

Clearly, π is monotone, and $\mathcal{F} = \{X \mid X \subseteq E \text{ is a minimal set satisfying } \pi(X) = 1\}$ is the family of all cut conjunctions of G.

In Sect. 4 we use the following notation. Let U be a subset of vertices of G, let F be a subset of edges of G, and let $G' = (V', E')$ and $G'' = (V'', E'')$ denote subgraphs of G (i.e., $V', V'' \subseteq V$ and $E', E'' \subseteq E$). We denote by $G[U]$ a subgraph of G induced on the vertex set U. Then $G - U \stackrel{\text{def}}{=} G[V \setminus U]$ is a graph obtained from G by deleting all the vertices of U and their incident edges, $G - F \stackrel{\text{def}}{=} (V, E \setminus F)$ is obtained by deleting all the edges of F from E and $G - G' \stackrel{\text{def}}{=} G - V'$. We also define $G + U \stackrel{\text{def}}{=} (V \cup U, E)$, $G + F \stackrel{\text{def}}{=} (V, E \cup F)$, and $G' + G'' \stackrel{\text{def}}{=} (V' \cup V'', E' \cup E'')$.

4.1 Characterization of Cut Conjunctions

It will be convenient to define a *cut* to be a set of edges $E(G_1, \ldots, G_l) = \bigcup_{i \neq j} \{uv \in E : u \in G_i, v \in G_j\}$ where $G_1, \ldots, G_l$ are induced subgraphs of G such that their vertex sets partition V, and G_i is connected for each $i = 1, \ldots, l$.

Let $B = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ be a set of distinct source-sink pairs of G. A *B-cut* is a cut $E(G_1, \ldots, G_l)$ such that, for each i, s_i and t_i do not belong to the same G_j. If the set B is clear from the context we shall call the minimal B-cut a *cut conjunction*. The following characterization of cut conjunctions follows directly from their definition.

Proposition 9 *Let $E(G_1, G_2, \ldots, G_l)$ be a B-cut. Then, $E(G_1, G_2, \ldots, G_l)$ is a minimal B-cut if and only if for every $x, y \in \{1, \ldots, l\}$ with $x \neq y$, if there is an edge of G between G_x and G_y then there must exist a source-sink pair (s_i, t_i) with exactly one vertex in G_x and the other in G_y (see Fig. 2).*

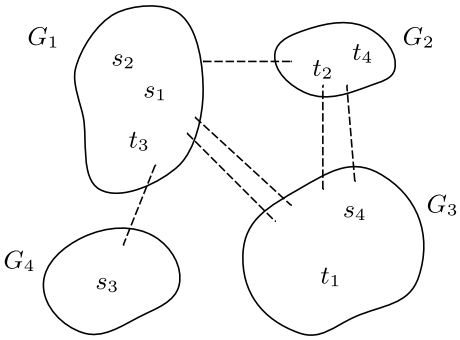

Fig. 2 Minimal B-cut $E(G_1, G_2, G_3, G_4)$. The *dashed lines* are the edges of the B-cut

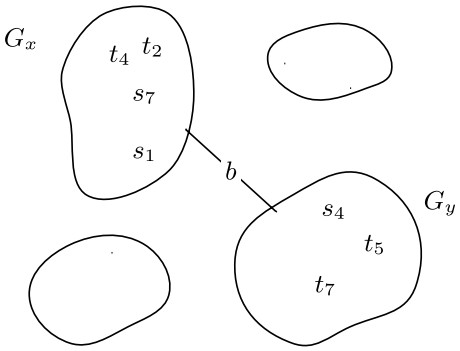

Fig. 3 Graph $G - (X \setminus b)$ contains two $(X \setminus b)$-conflicting pairs (s_4, t_4) and (s_7, t_7)

4.2 Reduction

In this section we reduce the problem of generating all elements of $\mathcal{Y}_{X,e}$ to generating all cut conjunctions in a graph of a simpler structure.

Let F be a subset of edges of G and let $(s_i, t_i) \in B$. Suppose that s_i and t_i are in the same component of $G - F$. Then we say that (s_i, t_i) is F-*conflicting*.

Let $X = E(G_1, G_2, \ldots, G_l)$ be a minimal B-cut of G and let $b \in X$. The removal of b from X reconnects some two components, G_x and G_y, of $G - X$, where one endpoint of b is in G_x and the other in G_y. Thus $G - (X \setminus b)$ contains at least one $(X \setminus b)$-conflicting pair (see Fig. 3). Hence generating all minimal sets $Y \subseteq E \setminus X$ which restore the property that no s_i is connected to t_i, is equivalent to generating all minimal B'-cuts in the graph $G_x + G_y + b$ where B' is the set of $(X \setminus b)$-conflicting pairs.

Let $L = G_x$ and $R = G_y$. We can always relabel the $(X \setminus b)$-conflicting pairs to guarantee that the conflicting s_i's are in L and the conflicting t_i's are in R. We denote the resulting graph by $H(X, b)$ (see Fig. 4). Note that we have reduced our generation problem to listing all cut conjunctions in $H(X, b)$. As we discuss in the next section, the latter problem can be efficiently solved by traversing an appropriately defined supergraph of cut conjunctions of $H(X, b)$.

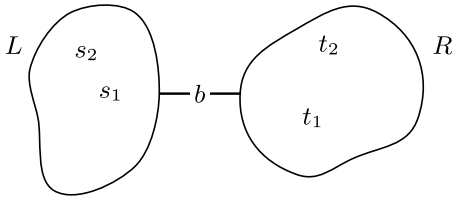

Fig. 4 Graph $H(X, b)$ with all sources in L and sinks in R

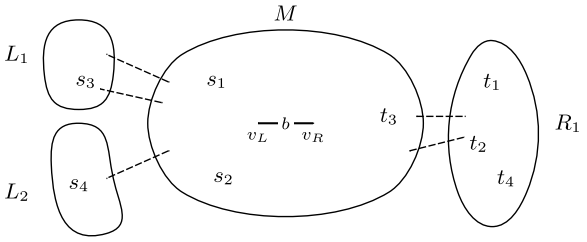

Fig. 5 Minimal B-cut $E(M, L_1, L_2, R_1)$. *Dashed lines* are the edges of the B-cut

4.3 Generating Cut Conjunctions in $H(X, b)$

In this section we describe an algorithm of generating cut conjunctions of the graph $H(X, b)$ defined at the end of Sect. 4.2. We apply the supergraph approach, i.e., we define the neighborhood operation on cut conjunctions of $H(X, b)$ so that the supergraph, whose vertices are these cut cunjunctions, is strongly connected, then we describe an algorithm of traversing this supergraph and analyze the complexity of the algorithm.

Let $H = H(X, b) = (V, E)$ be the graph defined at the end of Sect. 4.2, that is:

- $H = L + R + b$,
- $b = v_L v_R$ is a bridge (note that v_L can be a source and v_R can be a sink, but $b \neq s_i t_i$ for all i),
- L contains the sources $s_1, \ldots, s_{k'}$, and
- R contains the sinks $t_1, \ldots, t_{k'}$ (see Fig. 4).

Note that H is connected and the number k' of the vertex pairs in H is at most the number k of the vertex pairs in G.

4.3.1 Characterization of Cut Conjunctions of H

Let $B = \{(s_1, t_1), \ldots, (s_{k'}, t_{k'})\}$ and let $K = E(G_1, \ldots, G_l)$ be a cut conjunction of H, such that $K \neq \{b\}$. Without loss of generality, assume that b is in G_1. Note that every other G_j is contained either in L or in R (since G_j is connected and all paths from L to R go through b). We denote by $M = G_1$ the component containing b and call it the *root component of* K. The other components will be called *leaf components of* K. Denote the G_j's contained in L by $L_1, \ldots, L_m$ and those in R by $R_1, \ldots, R_n$ (see Fig. 5).

Proposition 10 *All edges of* $K = E(M, L_1, \ldots, L_m, R_1, \ldots, R_n)$ *lie between the root and leaf components. Hence* M *uniquely determines the leaf components of* K.

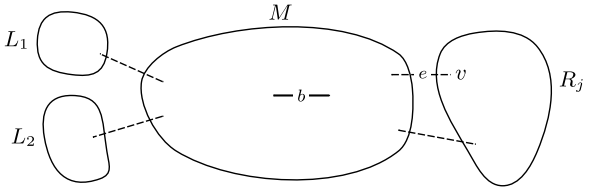

Fig. 6 Cut conjunction K in (q2-a)

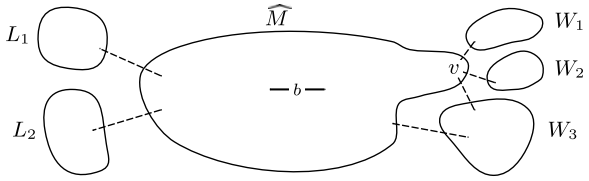

Fig. 7 B-cut D in (q2-a)

Proof Suppose that there is an edge $e \in K$ between two leaf components. Since there is no edge between L_i and R_j, we can assume that e connects L_i and L_j. But L_i and L_j contain only sources. Thus, by Proposition 9 , K is not minimal, a contradiction. □

4.3.2 Supergraph of Cut Conjunctions of H

Now we define the digraph $\mathcal{H}$, the *supergraph of cut conjunctions of* H. The vertex set of $\mathcal{H}$ is the family of all cut conjunctions of H other than $\{b\}$. For each cut conjunction $K = E(M, L_1, \ldots, L_m, R_1, \ldots, R_n)$ of H we define its out-neighborhood to consist of all cut conjunctions which can be obtained from K by the following sequence of steps (see example in Fig. 11):

(q1) Choose a vertex v incident to $e \in K$ such that $v \notin \{v_L, v_R\}$. Depending on v we have the following three cases.

(q2-a) Suppose v is in a leaf component of K and $M + v + e$ does not contain a source-sink pair (s_i, t_i). Without loss of generality, assume that $v \in R_j$ and either v is not a sink, or $v = t_i$ and $s_i \notin M$ (see Fig. 6).
Let $W_1, \ldots, W_p$ be the components of $R_j - v$, and let $\widehat{M} = M + v + \bigcup_{u \in M}\{uv \in E\}$. Then

$$D = E(\widehat{M}, L_1, \ldots, L_m, R_1, \ldots, R_{j-1}, W_1, \ldots, W_p, R_{j+1}, \ldots, R_n)$$

is a B-cut. Note that we have moved v from R_j to M. Removing v from R_j splits R_j into components $W_1, \ldots, W_p$ (see Fig. 7). In (q3) we may remove some edges of D to obtain a minimal B-cut.

(q2-b) Suppose v is in a leaf component of K and $M + v + e$ contains a source-sink pair (s_i, t_i). Without loss of generality, assume that $v \in R_j$ and $v = t_i$, $s_i \in M$ and $v_L \neq s_i$ (if $v_L = s_i$ we do not allow to include t_i to M). Let $W_1, \ldots, W_p$ be the components of $R_j - t_i$ and let $U_1, \ldots, U_r$ be the components of $M - s_i$ not containing b. Denote $\widehat{M} = (M + t_i + \bigcup_{u \in M}\{ut_i \in E\}) - (s_i + U_1 + \cdots + U_r)$. Then

$$D = E(\widehat{M}, L_1, \ldots, L_m, s_i, U_1, \ldots, U_r, R_1, \ldots, R_{j-1}, W_1, \ldots, W_p, R_{j+1}, \ldots, R_n)$$

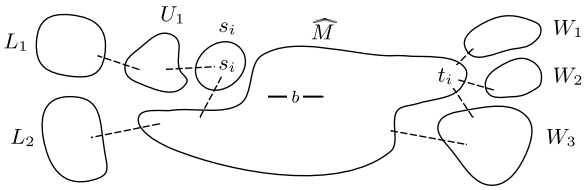

Fig. 8 B-cut D in (q2-b)

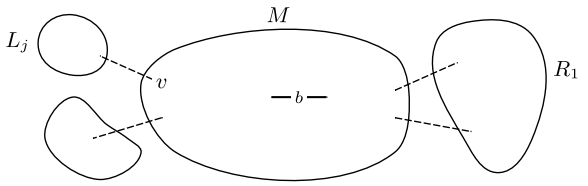

Fig. 9 Cut conjunction K in (q2-c)

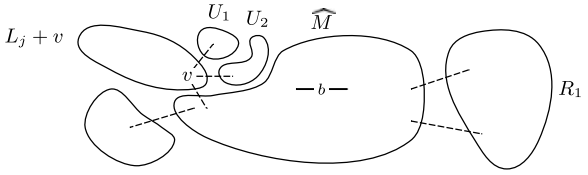

Fig. 10 B-cut D in (q2-c)

is a B-cut. Note that we have moved t_i from R_j to M. To restore the property that no s_i is connected to t_i, we have removed s_i from M. Removing v from R_j splits R_j into components $W_1, \ldots, W_p$, and removing s_i from M splits M into components $U_1, \ldots, U_r$ and $\widehat{M}$, the component containing b (see Fig. 8). In (q3) we may remove some edges of D to obtain a minimal B-cut.

(q2-c) Suppose $v \in M - \{v_L, v_R\}$. Without loss of generality, assume that v is adjacent to L_j (see Fig. 9). Note that $v \notin \{t_1, \ldots, t_{k'}\}$.

Let $U_1, \ldots, U_r$ be the components of $M - v$ not containing b, and let $\widehat{M} = M - (v + U_1 + \cdots + U_r)$. Then

$$D = E\Bigg(\widehat{M}, L_1, \ldots, L_{j-1}, L_j + v + \bigcup_{\substack{u \in L_j \\ uv \in E}} uv, L_{j+1}, \ldots, L_m, U_1, \ldots, U_r, R_1, \ldots, R_n\Bigg)$$

is a B-cut. Note that we have moved v from M to L_j splitting M into components $U_1, \ldots, U_r$ and $\widehat{M}$ (see Fig. 10). In (q3) we may remove some edges of D to obtain a minimal B-cut.

(q3) Let $D = E(G_1, \ldots, G_l)$ be the B-cut obtained in the previous step. Choose the lexicographically first two sets G_x and G_y such that there is an edge $e \in D$ connecting G_x and G_y and there is no $(D \setminus e)$-conflicting pair. Replace G_x and G_y in D by $G_x + G_y$. Repeat until no such pair exist, thus the resulting B-cut is minimal. Let $K' = E(M', L'_1, \ldots, L'_{m'}, R'_1, \ldots, R'_{n'})$ be the resulting cut conjunction. Then K' is a neighbor of K in $\mathcal{H}$.

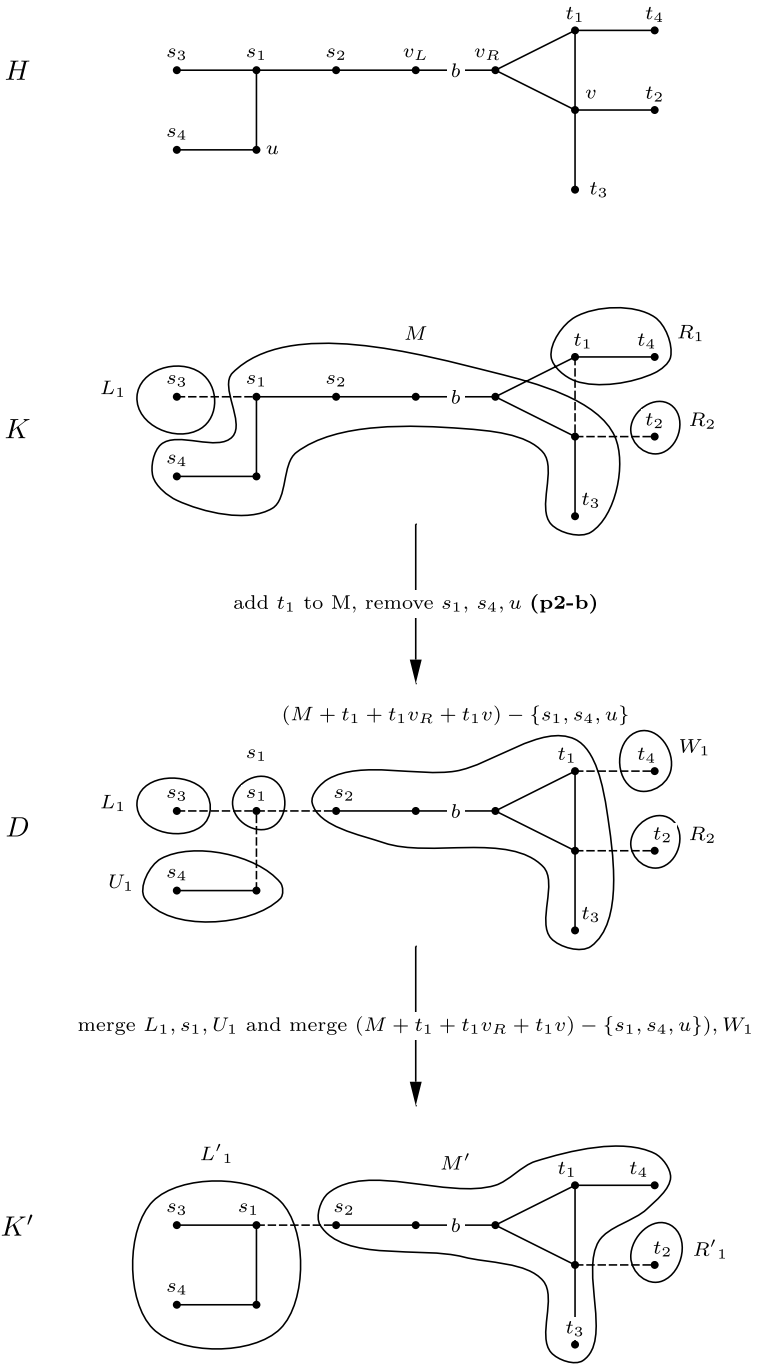

Fig. 11 Consider the graph H above and the cut conjunction $K = E(M, L_1, R_1, R_2) = E(\{s_1, s_2, s_4, u, v_L, v_R, t_3, v\}, \{s_3\}, \{t_1, t_4\}, \{t_2\})$. Then $K' = E(M', L'_1, R'_1) = E(\{s_2, v_L, v_R, t_1, t_3, t_4, v\}, \{s_1, s_3, s_4, u\}, \{t_2\})$ is a neighbor of K obtained by moving t_1 to M

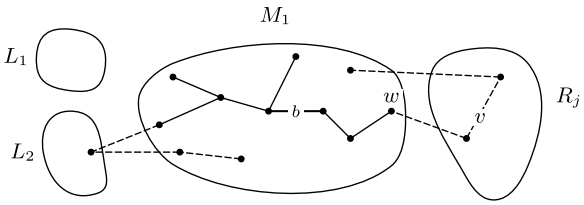

Fig. 12 Cut conjunction K_1. *Solid lines* are the K_1-solid edges, *dashed lines* are the K_1-dashed edges

Proposition 11 *The supergraph $\mathcal{H}$ is strongly connected.*

To prove Proposition 11 we need two lemmas.

Let K_1, K_3 be cut conjunctions and let M_1, M_3 be their root components. We call the vertices of M_3 *blue vertices*, and all other vertices *green vertices*. Let $\mathcal{K}$ be an induced subgraph of $\mathcal{H}$, whose vertices are the cut conjunctions with root components containing all the blue vertices. Note that $\mathcal{K}$ has at least one vertex, namely K_3.

Lemma 12 *There exists a cut conjunction $K_2 \in \mathcal{K}$ such that there is a path from K_1 to K_2 in $\mathcal{H}$.*

Proof Let T be an arbitrary spanning tree of M_3 containing the bridge b. For a B-cut D of H with M as its root component, we partition the edges of T into two groups. Edges that form a contiguous part within M will be called D-solid edges, and the remaining edges will be called D-dashed edges. More precisely, we call an edge e of T a *D-solid edge*, if

- $e \in M$,
- e is reachable from b by using only edges of T that are in M.

Otherwise e is called a *D-dashed edge* (see Fig. 12). Note that b is a D-solid edge. We denote the set of D-solid edges by S_D and the set of D-dashed edges by D_D. Clearly, $|S_D| + |D_D| = |T|$.

Let $K_1 = E(M_1, L_1, \ldots, L_m, R_1, \ldots, R_n^1)$. We will show by induction on the number of K_1-solid edges $|S_{K_1}|$ that there is a path from K_1 to K_2.

If $|S_{K_1}| = |T|$, then M_1 contains the spanning tree T of blue vertices. Hence $K_1 \in \mathcal{K}$.

If $|S_{K_1}| < |T|$, then there exists a K_1-dashed edge vw between two blue vertices v and w such that v is in a leaf component of K_1, $w \in M_1$ and w is incident to a K_1-solid edge. Without loss of generality, suppose that $v \in R_j$ (see Fig. 12). Such an edge exists because K_1-dashed and K_1-solid edges form the spanning tree of blue vertices.

We now show that K_1', a neighbor of K_1, obtained by moving the blue vertex v from the leaf to the root component, has $|S_{K_1'}| \geq |S_{K_1}| + 1$. Depending on v there are two cases.

Case 1: v is not a sink or $v = t_i$ and $s_i \notin M_1$. Let D be the B-cut obtained in (q2-a) and M_D be its root component. Recall that $M_D = M_1 + v + \bigcup_{u \in M}\{uv \in E\}$. Thus S_D contains all K_1-solid edges. Since M_D contains both v and w, vw is a D-solid edge, so $|S_D| = |S_{K_1}| + 1$. In (q3) M_D can only merge with leaf components, hence $|S_{K_1'}| \geq |S_D|$. This implies that $|S_{K_1'}| \geq |S_{K_1}| + 1$.

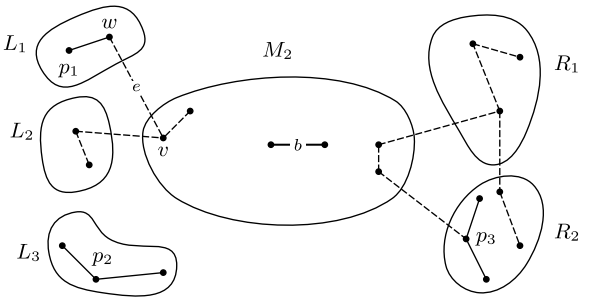

Fig. 13 Cut conjunction K_2. The *solid lines* are K_2-solid edges, the *dashed lines* are K_2-dashed edges

Case 2: $v = t_i$, $s_i \in M$. Note that t_i is a blue vertex, so s_i must be green, since M_3 does not contain any source-sink pair, and in particular s_i cannot be an endpoint of b. Let D be the B-cut obtained in (q2-b) and M_D be its root component. Recall that $M_D = (M_1 + t_i + \bigcup_{u \in M}\{ut_i \in E\}) - (s_i + U_1 + \cdots + U_r)$, where $U_1, \ldots, U_r$ are the components of $M - s_i$ not containing b.

Observe that in (q2-b) we did not remove any K_1-solid edge from M_1. Since s_i is a green vertex, all edges incident to s_i do not belong to T. Edges in $U_1, \ldots, U_r$ and incident to these components are also not K_1-solid, because all paths from b to $U_1, \ldots, U_r$, which use edges of T that are in M_1, must go through s_i. Thus $|S_D| = |S_{K_1}| + 1$.

In (q3) M_D can only increase its size after merging with leaf components, hence $|S_{K_1'}| \geq |S_D|$. This implies that $|S_{K_1'}| \geq |S_{K_1}| + 1$. □

Lemma 13 *For every $K_2 \in \mathcal{K}$ there is a path from K_2 to K_3 in $\mathcal{K}$.*

Proof Let $W_1, \ldots, W_q$ be the leaf components of K_3 and $T_1, \ldots, T_q$ be arbitrary spanning trees of $W_1, \ldots, W_q$. Recall that vertices of $W_1, \ldots, W_q$ are called green vertices.

For every leaf W_j there is at least one source-sink pair (s_i, t_i) such that one of s_i and t_i belongs to W_j and the other to the root component of K_3. Choose one such source or sink for every W_j and denote this set by $P = \{p_1, \ldots, p_q\}$.

Let $D = E(M, G_1, \ldots, G_l)$ be a B-cut of H such that all vertices of P are in the leaf components. Let $e \in T_i$ for some $i \in \{1, \ldots, q\}$. We call e a *D-solid edge* if there is $j \in \{1, \ldots, l\}$ such that $e \in G_j$, $p_i \in G_j$ and e is reachable from p_i by using only edges of T_i that are in G_j. Otherwise e is called a *D-dashed edge* (see Fig. 13). We denote the set of D-solid edges by S_D and the set of D-dashed edges by D_D. Note that $|S_D| + |D_D| = |T_1| + \cdots + |T_q|$.

Let $K_2 = E(M_2, L_1, \ldots, L_m, R_1, \ldots, R_n)$. Recall that M_3 is the root component of K_3 and its vertices are called blue vertices. Since $M_3 \subseteq M_2$, all elements of P must belong to leaf components of K_2 and thus the notion of K_2-solid edges is well defined. We will show by induction on the number of K_2-solid edges $|S_{K_2}|$, that there is a path in $\mathcal{K}$ from K_2 to K_3 (note that since this path is in $\mathcal{K}$, the root components of vertices on that path must contain all the blue vertices).

If $|S_{K_2}| = |T_1| + \cdots + |T_q|$, all green vertices are in leaf components, so M_2 contains only blue vertices, thus $M_2 = M_3$ and by Proposition 10, we have $K_2 = K_3$.

If $|S_{K_2}| < |T_1| + \cdots + |T_q|$, then there exists a K_2-dashed edge $e = vw$ between two green vertices v and w such that w is in a leaf component, $v \in M_2$ and w is incident to a K_2-solid edge or $w = p_i$. Without loss of generality, suppose that $e \in T_i$ and $w \in L_j$ (see Fig. 13). Such an edge exists because K_2-dashed and K_2-solid edges form a spanning forest of green vertices.

We show that K_2', a neighbor of K_2 obtained by moving v from M_2 to L_j, has $|S_{K_2'}| \geq |S_{K_2}| + 1$ and $K_2' \in \mathcal{K}$.

Let $D = E(\widehat{M}, L_1, \ldots, L_j + v, \ldots, L_m, U_1, \ldots, U_r, R_1, \ldots, R_n)$ be the B-cut obtained in (q2-c). Recall that $\widehat{M} = M_2 - (v + U_1 + \cdots + U_r)$, where $U_1, \ldots, U_r$ are the components of $M_2 - v$ not containing b. Note also that $U_1, \ldots, U_r$ cannot contain any blue vertices, since M_2 contains M_3, which is connected, thus removing a green vertex v cannot disconnect any blue vertex from b. Hence $M_3 \subseteq \widehat{M}$. Since in (q3) $\widehat{M}$ can only increase its size, the root component of K_2' contains M_3.

Since $L_j + v$ contains both v and w, e is a D-solid edge. Thus $|S_D| = |S_{K_2}| + 1$. In (q3) only leaf components not containing vertices of P can merge with $\widehat{M}$. Since these leaf components do not contain any solid edges, we obtain $|S_{K_2'}| \geq |S_D|$. This implies that $|S_{K_2'}| \geq |S_{K_2}| + 1$. □

Proof of Proposition 11 Let K_1 and K_3 be arbitrary cut conjunctions and $\mathcal{K}$ be the induced subgraph of $\mathcal{H}$ defined above. By Lemma 12 there is a path in $\mathcal{H}$ from K_1 to some cut conjunction K_2 in $\mathcal{K}$. By Lemma 13 there is a path from any cut conjunction of $\mathcal{K}$ to K_3. The proposition follows. □

4.3.3 Algorithm of Generating Cut Conjunctions of H

Since $\mathcal{H}$ is strongly connected we can generate all cut conjunctions of H by performing a breadth-first search in $\mathcal{H}$. Recall that a root component uniquely determines the

Traversal($\mathcal{H}$)

Find an initial root component M^0: $M^0 \leftarrow \{v_L, v_R\}$, repeat adding adjacent and nonconflicting vertex to M^0 until no such vertex exists.
Initialize a queue $\mathcal{Q} \leftarrow \emptyset$ and a dictionary of visited vertices $\mathcal{D} \leftarrow \emptyset$.
Perform a breadth-first search of $\mathcal{H}$ starting from M^0:

1 **output** the cut conjunction X^0 corresponding to M^0, insert M^0 to $\mathcal{Q}$ and to $\mathcal{D}$
2 **while** $\mathcal{Q} \neq \emptyset$ **do**
3 take the first vertex M out of the queue $\mathcal{Q}$
4 find the sets N_1 and N_2 of vertices adjacent to M and $V \setminus M$
6 **for** every vertex $v \in N_1 \cup N_2$ **do**
7 **if** $v \in N_1$ **then** $M' \leftarrow M \cup v$
else M' is the set of vertices reachable from v_L in $H[M \setminus v]$
8 add adjacent and nonconflicting vertex to M', repeat until no such vertex exists
9 **if** $M' \notin \mathcal{D}$ **then**
output M' corresponding to M', insert it to $\mathcal{Q}$ and to $\mathcal{D}$

cut conjunctions of H. Thus we generate root components but we output the corresponding cut conjunctions.

We say that a vertex is *nonconflicting to a root component* M if $M \cup v$ does not contain a source-sink pair.

Proposition 14 *Traversal*($\mathcal{H}$) *generates* K (*or all*) *cut conjunctions of* H *in* $O(K \log(K)nm)$ *time, where* n *and* m *are the number of vertices and edges of* H, *respectively.*

Proof Since H is connected, we have $n \leq m$.

We assume that we have a binary vector of length k' associated to a root component indicating that the root component contains the ith source or sink. Thus we can test if a vertex is nonconflicting to a root component in $O(1)$ time.

Finding an initial root component M^0 using a breadth-first search takes in $O(m)$ time.

Since a vertex is removed from $\mathcal{Q}$ every time we execute the while loop and it will never be reinserted to $\mathcal{Q}$, the while loop is executed at most K times. Note that computing sets N_1 and N_2 takes $O(m)$ time and $|N_1 \cup N_2| \leq n$. Thus we perform the for loop at most n times.

Computing M' takes $O(m)$ time, checking if M' is in the dictionary takes $O(\log(K)m)$ (we implement $\mathcal{D}$ as a balanced binary search tree) and finding the cut conjunction corresponding to M' takes $O(m)$ time.

Thus *Transversal*($\mathcal{H}$) generates K (or all) cut conjunctions in $O(K \log(K)nm)$ time. □

4.4 Complexity

In this section we utilize Proposition 8 to analyze the total running time of the procedure *Transversal*($\mathcal{G}$). Let $n = |V|$, $m = |E|$.

Since using a breadth-first search one can test if an edge set is a cut conjunction in $O(k(n+m))$ time, we have $\gamma(E) = O(k(n+m))$. As H has at most n vertices and m edges and by Proposition 14 we obtain $\phi(K, E) = O(K \log(K)nm)$.

By Proposition 8 procedure *Transversal*($\mathcal{G}$) generates K (or all) cut conjunctions in $O(K^2 \log(K)nm^2 + K^2k(n+m)m^2)$ time. This completes the proof of Theorem 1.

5 Proof of Theorem 2

We apply the $X - e + Y$ method to the generation of all minimal bridge-avoiding extensions.

It will be convenient to assume in this section that the input graph $G = (V, E)$ may contain parallel edges, i.e., that G is a *multigraph*. For a nonempty set $B \subseteq E$ we define a Boolean function π as follows: for a subset $X \subseteq E \setminus B$

$$\pi(X) = \begin{cases} 1, & \text{no } b \in B \text{ is a bridge in } (V, X \cup B); \\ 0, & \text{otherwise.} \end{cases}$$

Clearly π is monotone. Then $\mathcal{F} = \{X \mid X \subseteq E \setminus B$ is a minimal set satisfying $\pi(X) = 1\}$ is the family of all minimal bridge-avoiding extensions of B.

We show that generating elements of $\mathcal{Y}_{X,e}$ is equivalent to generating all directed paths between a pair of vertices in some explicitly given directed multigraph.

Let $X \in \mathcal{F}$ and $e \in X$. We define $B' = \{b_1, \ldots, b_k\}$ to be the subset of edges of B that are bridges in $(V, B \cup (X \setminus e))$.

Claim 15 *There is a path in* $(V, B \cup (X \setminus e))$ *containing all edges of* B'.

Proof First observe that for each edge $b_i \in B'$ there is a cycle C_i in $(V, B \cup X)$ containing e and b_i. Suppose $b_i \in C_i \setminus C_j$ for some $i, j \in \{1, \ldots, k\}$. Then there is a cycle C' consisting of some edges of C_i and C_j such that $b_i \in C'$ and $e \notin C'$. Note that C' is also the cycle in $(V, B \cup (X \setminus e))$. Thus b_i is not a bridge in $(V, B \cup (X \setminus e))$, a contradiction. Hence there is a cycle in $(V, B \cup X)$ containing $\{b_1, \ldots, b_k\}$ and e, and consequently, edges of this cycle without e form a path in $(V, B \cup (X \setminus e))$. □

We next construct the multigraph $G' = (V', E')$ from $(V, E \setminus e)$ by contracting all edges in $(B \setminus B') \cup (X \setminus e)$. By Claim 15 the edges of B' form a path in G'. Note that edges if G and G' are in the one to one correspondence. Moreover no edge $b \in B$ is a bridge in $(V, B \cup (X \setminus e) \cup Y)$, where $Y \subseteq E \setminus (X \cup B)$, if and only if no edge $b \in B'$ is a bridge in $(V', B' \cup Y')$, where Y' is the set of edges of G' corresponding to Y. Thus the general generation problem for cut conjunctions in cocycle matroids reduces to the special case of the same problem for multigraphs in which B is a path.

Let $u_1, \ldots, u_{k+1}$ denote the $k+1$ vertices on the path $B' = \{b_1, \ldots, b_k\}$ in G'. We can assume without loss of generality that $b_i = u_i u_{i+1}$ for $i = 1, \ldots, k$. We next consider the directed multigraph $\widetilde{G}' = (V', \widetilde{E}')$ obtained from the multigraph $G' = (V', E')$ by replacing the undirected path B' by the directed path $\widetilde{B}' = u_1 \leftarrow u_2 \leftarrow \cdots \leftarrow u_k \leftarrow u_{k+1}$ and by adding two opposite arcs $u \to v$ and $v \to u$ for each of the remaining edges $uv \in E' \setminus B'$.

We show that no edge $b \in B'$ is a bridge in $(V', B' \cup Y')$, where $Y' \subseteq E' \setminus B'$ if and only if there is a u_1–u_{k+1} dipath corresponding to Y' in $\widetilde{G}'$.

If no edge $b \in B'$ is a bridge in $(V', B' \cup Y')$, then for each $i = 1, \ldots, k$ there must exist a path $P \subseteq Y'$ such that

($\mathcal{P}'$) P and B' are edge disjoint and
($\mathcal{P}''$) the vertex set of P contains exactly two vertices u_α, u_β of B' such that $\alpha \le i$ and $\beta \ge i+1$.

By the minimality of Y' we conclude that

$$Y' = P_1 \cup \cdots \cup P_s \tag{1}$$

for some paths $P_1, \ldots, P_s$ satisfying conditions ($\mathcal{P}'$) and ($\mathcal{P}''$) above, where no two distinct paths in the above decomposition have a common vertex outside of B'. Denoting by u_{α_i} and u_{β_i} the intersection of the vertex set of P_i with B', we can also assume without loss of generality that

$$u_1 = \alpha_1 < \alpha_2 \le \beta_1 < \alpha_3 \le \beta_2 < \alpha_4 \le \cdots < \alpha_s \le \beta_{s-1} < \beta_s = u_{k+1}, \tag{2}$$

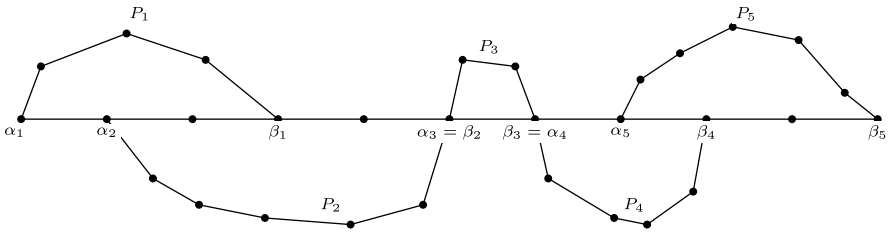

Fig. 14 Subgraph $(V', B' \cup Y')$

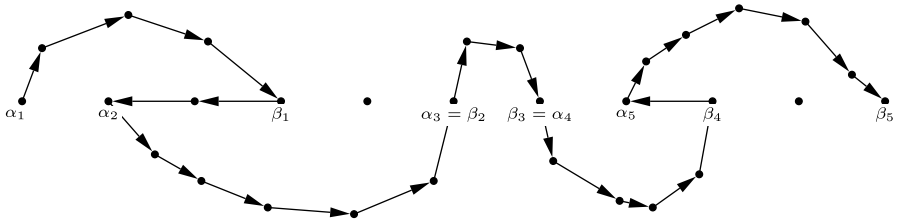

Fig. 15 Directed path in $\widetilde{G'}$

where some pairs of consecutive paths P_j and P_{j+1} may have the same endpoint on B' (see Fig. 14).

From the above discussion it follows that there exists a one to one correspondence between all minimal sets Y' admitting decomposition (1) which satisfies (2) and all directed paths from u_1 to u_{k+1} in $\widetilde{G'}$ (see Fig. 15).

We next utilize Proposition 8 to analyze the total running time of the procedure *Transversal*$(\mathcal{G})$. Let n and m be the number of vertices and edges of G, respectively.

Since one can find all bridges in G in $O(n+m)$ time, we have $\gamma(E) = O(n+m)$ [15]. Note that contracting an edge takes $O(n+m)$ time, thus we can construct $\widetilde{G'}$ in $O(m(n+m))$ and $\widetilde{G'}$ has at most n vertices and $2m$ arcs. As K paths between a given pair of vertices can be generated via backtracking in $O(Km+n+m)$ time [11], we obtain $\phi(K, E) = Km + m(n+m)$. By Proposition 8 the procedure *Transversal*$(\mathcal{G})$ generates K (or all) minimal bridge-avoiding extensions in $O(K^2 \log(K) m^2 + K^2 m^2 (n+m))$ time. This completes the proof of Theorem 2.

6 Proof of Proposition 4

Let us consider a binary matroid M on ground set $S = A \cup B$, where $B = \{b_1, b_2\}$. As we mentioned in the Introduction, it is enough to consider the dual formulation of the cut conjunction problem:

> *Generate all minimal subsets* $X \subseteq A \stackrel{\text{def}}{=} S \setminus B$ *such that* $X \cup \{b_2\}$ *spans* b_1 *and* $X \cup \{b_1\}$ *spans* b_2 *in the dual matroid* M^*.

To see that this generation problem is tractable, we show first that for a subset X of A, b_1 is a linear combination of vectors of $X \cup \{b_2\}$ and b_2 is a linear combination of

vectors of $X \cup \{b_1\}$ if and only if $b_1 + b_2$ is a linear combination of vectors of X.

$$\text{If} \quad \sum_{a \in Y} a = b_1 + b_2, \quad \text{where } Y \subseteq X, \quad \text{then} \quad \sum_{a \in Y} a + b_1 = b_2$$
$$\text{and} \quad \sum_{a \in Y} a + b_2 = b_1.$$

Conversely, we consider $X \subseteq A$ such that b_1 is a linear combination of $X \cup \{b_2\}$ and b_2 is a linear combination of $X \cup \{b_1\}$. Depending on whether these linear combination include b_2 and b_1, respectively, we have two cases:

Case 1: b_2, b_1 do not appear in either of the linear combinations. Thus $\sum_{a \in X_1} a = b_1$, $\sum_{a \in X_2} a = b_2$, where $X_1, X_2 \subseteq X$. Then $\sum_{a \in (X_1 \cup X_2) \setminus (X_1 \cap X_2)} a = b_1 + b_2$.

Case 2: suppose b_2 appears in the first linear combination. Thus $\sum_{a \in Y} a + b_2 = b_1$, where $Y \subseteq X$. Then $\sum_{a \in Y} a = b_1 + b_2$.

Hence X is a minimal subset of A such that $X \cup \{b_2\}$ spans b_1 and $X \cup \{b_1\}$ spans b_2 in M^* if and only if X is a minimal subset of A spanning $b_1 + b_2$ in the matroid on ground set $A \cup \{b_1 + b_2\}$. Thus our problem reduces to the generation of all circuits containing $b_1 + b_2$ in the matroid on ground set $A \cup \{b_1 + b_2\}$, which can be done in incremental polynomial time [2].

Remark 16 A similar simplification does not work for $|B| > 2$. For instance, for $B = \{b_1, b_2, b_3\}$, the facts that b_i is a linear combination of vectors of $X \cup \{B \setminus b_i\}$, for $i = 1, 2, 3$, do not imply that $b_1 + b_2 + b_3$ is a linear combination of vectors of X. Consider e.g., the vectors $a_1 = (1, 1, 0, 0)$, $a_2 = (1, 0, 1, 0)$, $a_3 = (0, 1, 1, 0)$, $b_1 = (1, 0, 0, 1)$, $b_2 = (0, 1, 0, 1)$ and $b_3 = (0, 0, 1, 1)$ in 4-dimensions (mod 2) satisfying $b_1 = a_1 + b_2 = a_2 + b_3$, $b_2 = a_1 + b_1 = a_3 + b_3$, $b_3 = a_2 + b_1 = a_3 + b_2$, but $b_1 + b_2 + b_3 = (1, 1, 1, 1)$ is not in the linear space spanned by a_1, a_2, and a_3.

Appendix: Proof of Proposition 3

For the sake of completeness we present the proof of Proposition 3. An alternative proof can be found in [2].

Let M be a vectorial matroid on ground set S, let $B \subseteq S$ and let $\mathcal{F}$ be the family of all maximal subsets of $A \stackrel{\text{def}}{=} S \setminus B$ that span no vector $b \in B$. In this section we show that given a subfamily $\mathcal{X} \subseteq \mathcal{F}$, it is NP-hard to decide whether $\mathcal{X} \neq \mathcal{F}$. We reduce our problem from the well known 3-satisfiability.

Let $\phi = C_1 \wedge C_2 \cdots \wedge C_m$ be a given CNF on n variables with exactly three literals per clause. We may represent the sets A and B as matrices. We let

$$A = (\vec{a}^{\bar{x}_1}, \vec{a}^{\bar{x}_2}, \ldots, \vec{a}^{\bar{x}_n}, \vec{a}^{x_1}, \vec{a}^{x_2}, \ldots, \vec{a}^{x_n}),$$

where $\vec{a}^{\bar{x}_i}$ and $\vec{a}^{x_i}$ are $(n+1)$-dimensional vectors defined as

$$(\vec{a}^{\bar{x}_i})_j = \begin{cases} 1, & \text{if } i = j; \\ 0, & \text{otherwise,} \end{cases} \qquad (\vec{a}^{x_i})_j = \begin{cases} 1, & \text{if } i = j \text{ or } i = n+1; \\ 0, & \text{otherwise.} \end{cases}$$

For every clause $C_p = l_{i_1} \vee l_{i_2} \vee l_{i_3}$, where $l_{i_j} \in \{x_{i_j}, \bar{x}_{i_j}\}$, and $\alpha \in \{0, \ldots, n-3\}$, we define

$$\vec{b}^{p,\alpha} = 4n\vec{a}^{l_{i_1}} + 2n\vec{a}^{l_{i_2}} + n\vec{a}^{l_{i_3}} + \vec{f}^{p} + \alpha\vec{e},$$

where $\vec{f}^{p}$ and $\vec{e}$ are $(n+1)$-dimensional vectors defined as

$$(\vec{f}^{p})_i = \begin{cases} 0, & \text{if } i \in \{i_1, i_2, i_3, n+1\}; \\ 1, & \text{otherwise,} \end{cases} \quad \text{and} \quad \vec{e} = (0, \ldots, 0, 1)^T.$$

Then $B = (\vec{b}^{p,\alpha})$, for $p = 1, \ldots, m$ and $\alpha = 0, \ldots, n-3$ (see Example 17).

Example 17 Consider $\phi = C_1 \wedge C_2 = (x_1 \vee \bar{x}_2 \vee x_3)(x_1 \vee \bar{x}_4 \vee \bar{x}_5)$. Then $A = (\vec{a}^{\bar{x}_1}, \vec{a}^{\bar{x}_2}, \ldots, \vec{a}^{\bar{x}_5}, \vec{a}^{x_1}, \vec{a}^{x_2}, \ldots, \vec{a}^{x_5})$, i.e.,

$$A = \left(\begin{array}{ccccc|ccccc} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{array}\right)$$

and $B = \{\vec{b}^{1,0}, \vec{b}^{1,1}, \vec{b}^{1,2}, \vec{b}^{2,0}, \vec{b}^{2,1}, \vec{b}^{2,2}\}$, where

$$\vec{b}^{1,\alpha} = \begin{pmatrix} 4 \cdot 5 \\ 2 \cdot 5 \\ 5 \\ 1 \\ 1 \\ \hline 4 \cdot 5 + 5 + \alpha \end{pmatrix} \quad \text{and} \quad \vec{b}^{2,\alpha} = \begin{pmatrix} 5 \\ 1 \\ 1 \\ 4 \cdot 5 \\ 2 \cdot 5 \\ \hline 5 + \alpha \end{pmatrix}.$$

Claim 18 *For each $i \in \{1, \ldots, n\}$, $A \setminus \{\vec{a}^{\bar{x}_i}, \vec{a}^{x_i}\}$ is a maximal subset of A spanning no $\vec{b} \in B$.*

Proof Observe that all vectors of $A \setminus \{\vec{a}^{\bar{x}_i}, \vec{a}^{x_i}\}$ have ith entry zero and every $\vec{b} \in B$ has all entries nonzero. Both $A \setminus \{\vec{a}^{\bar{x}_i}\}$ and $A \setminus \{\vec{a}^{x_i}\}$ span all $b \in B$, since $\operatorname{rank}(A \setminus \{\vec{a}^{\bar{x}_i}\}) = \operatorname{rank}(A \setminus \{\vec{a}^{x_i}\}) = n + 1$. Thus $A \setminus \{\vec{a}^{\bar{x}_i}, \vec{a}^{x_i}\}$ is maximal subset of A spanning no $\vec{b} \in B$. □

Let $\mathcal{X} = \{A \setminus \{\vec{a}^{\bar{x}_1}, \vec{a}^{x_1}\}, \ldots, A \setminus \{\vec{a}^{\bar{x}_n}, \vec{a}^{x_n}\}\} \subseteq \mathcal{F}$. We shall call elements of $\mathcal{F} \setminus \mathcal{X}$ *nontrivial*. Let $\mathcal{H}$ be a family of subsets of A of the form $(\vec{a}^{l_1}, \vec{a}^{l_2}, \ldots, \vec{a}^{l_n})$, where $l_i \in \{x_i, \bar{x}_i\}$, i.e. subsets of A that contain exactly one of each pair $\vec{a}^{\bar{x}_i}, \vec{a}^{x_i}$, for $i \in \{1, \ldots, n\}$.

Claim 19 *Every nontrivial element X of $\mathcal{F}$ belongs to $\mathcal{H}$.*

Proof X is a maximal subset of A spanning no $\vec{b} \in B$ and is not a subset of an element of $\mathcal{X}$, thus X must contain at least one of each pair $\vec{a}^{\bar{x}_i}, \vec{a}^{x_i}$. Suppose that for some j, X contains both $\vec{a}^{\bar{x}_j}, \vec{a}^{x_j}$. Then $\operatorname{rank}(X) = n + 1$, thus X spans all $\vec{b} \in B$, a contradiction. Hence X contains exactly one of $\vec{a}^{\bar{x}_i}, \vec{a}^{x_i}$, for $i \in \{1, \ldots, n\}$. □

Now let $X = (\vec{a}^{l_1}, \vec{a}^{l_2}, \ldots, \vec{a}^{l_n}) \in \mathcal{H}$ and $\vec{x} = (x_1, \ldots, x_n)$ be an assignment of ϕ. We define a bijection between elements of $\mathcal{H}$ and assignments of ϕ as follows: $x_i = 0$ if and only if $\vec{a}^{x_i} \in X$, $x_i = 1$ if and only if $\vec{a}^{\bar{x}_i} \in X$.

Claim 20 *X is nontrivial element of $\mathcal{F}$ if and only if $\vec{x}$ is a satisfying assignment of ϕ.*

Proof Let X be nontrivial element of $\mathcal{F}$. By Claim 19, $X \in \mathcal{H}$, so there exists an assignment $\vec{x}$ corresponding to X. Suppose that $\vec{x}$ is not a satisfying assignment, then $\vec{x}$ does not satisfy a clause $C_p = l_{i_1} \vee l_{i_2} \vee l_{i_3}$. Thus $l_{i_1}, l_{i_2}, l_{i_3}$ are assigned 0. Then $\{\vec{a}^{l_{i_1}}, \vec{a}^{l_{i_2}}, \vec{a}^{l_{i_3}}\} \in X$. Let $\alpha = \sum_{j \notin \{i_1, i_2, i_3\}} (1 - x_j)$ be the number of 0's in entries of $\vec{x}$ different than i_1, i_2, i_3.

Then $\sum_{i \notin \{i_1, i_2, i_3\}} \vec{a}^{l_i} = \vec{f} + \alpha \vec{e}$, hence $\vec{b}^{p,\alpha} = 4n\vec{a}^{l_{i_1}} + 2n\vec{a}^{l_{i_2}} + n\vec{a}^{l_{i_3}} + \sum_{i \notin \{i_1, i_2, i_3\}} \vec{a}^{l_i}$. Thus $\vec{b}^{p,\alpha}$ is spanned by X, a contradiction (see Example 21).

Now let $\vec{x}$ be a satisfying assignment. We will show that X spans no $b \in B$. Choose $\vec{b}^{p,\alpha} = (b_1, \ldots, b_{n+1}) \in B$ corresponding to the clause $C_p = l_{i_1} \vee l_{i_2} \vee l_{i_3}$. Observe that $X = \left(\frac{I_n}{r}\right)$, where I_n is $n \times n$ identity matrix and $\vec{r} = (r_{l_1}, \ldots, r_{l_n})$ is a n-dimensional vector. Then the system $I_n \vec{y} = (b_1, \ldots, b_n)$ has a unique solution

$$y_i = b_i = \begin{cases} 4n, & \text{if } i = i_1; \\ 2n, & \text{if } i = i_2; \\ n, & \text{if } i = i_3; \\ 1, & \text{otherwise.} \end{cases}$$

However the linear combination, with coefficients y_i, of entries of the last row of A cannot be equal to b_{n+1}, the last entry of $\vec{b}^{p,\alpha}$, for any $\alpha \in \{0, \ldots, n-3\}$ (see Example 22), because

- the linear combination is $\sum_{i=1\ldots n} y_i r_{l_i} = 4nr_{i_1} + 2nr_{i_2} + nr_{i_3} + \beta$, where $\beta = \sum_{i \notin \{i_1, i_2, i_3\}} (1 - x_i)$ is the number of zero entries of $\vec{x}$ different than i_1, i_2, i_3,
- $b_{n+1} = 4n(\vec{a}^{l_{i_1}})_{n+1} + 2n(\vec{a}^{l_{i_2}})_{n+1} + n(\vec{a}^{l_{i_3}})_{n+1} + \alpha$,
- there is at least one index j of $\{i_1, i_2, i_3\}$ such that it satisfies $(\vec{a}^{l_j})_{n+1} \neq r_j$ (since $\vec{x}$ is a satisfying assignment, it must satisfy every clause).

Hence X is nontrivial element of $\mathcal{F}$. □

Example 21 Let ϕ, A, B be as defined in Example 17. A nonsatisfying assignment $\vec{x} = (0, 1, 0, 0, 1)$ of ϕ corresponds to

$$X = (\vec{a}^{x_1}, \vec{a}^{\bar{x}_2}, \vec{a}^{x_3}, \vec{a}^{x_4}, \vec{a}^{\bar{x}_5}) = \left(\begin{array}{ccccc} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ \hline 1 & 0 & 1 & 1 & 0 \end{array}\right).$$

$\vec{x}$ does not satisfy the first clause $x_1 \vee \bar{x}_2 \vee x_3$, number of 0's not in the first, second or third entry of $\vec{x}$ is 1, thus X spans $\vec{b}^{1,1}$:

$$4 \cdot 5 \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \hline 1 \end{pmatrix} + 2 \cdot 5 \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ \hline 0 \end{pmatrix} + 5 \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \hline 1 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ \hline 1 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ \hline 0 \end{pmatrix} = \begin{pmatrix} 4 \cdot 5 \\ 2 \cdot 5 \\ 5 \\ 1 \\ 1 \\ \hline 4 \cdot 5 + 5 + 1 \end{pmatrix}.$$

Example 22 A satisfying assignment $\vec{x} = (1, 0, 0, 0, 1)$ of ϕ corresponds to

$$X = (\vec{a}^{\bar{x}_1}, \vec{a}^{x_2}, \vec{a}^{x_3}, \vec{a}^{x_4}, \vec{a}^{\bar{x}_5}) = \left(\begin{array}{c} I_5 \\ \hline \vec{r} \end{array}\right) = \left(\begin{array}{ccccc} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 1 & 1 & 1 & 0 \end{array}\right).$$

Choose

$$\vec{b}^{1,\alpha} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ \hline b_6 \end{pmatrix} = \begin{pmatrix} 4 \cdot 5 \\ 2 \cdot 5 \\ 5 \\ 1 \\ 1 \\ \hline 4 \cdot 5 + 5 + \alpha \end{pmatrix}$$

corresponding to the first clause $x_1 \vee \bar{x}_2 \vee x_3$. Then the system $I_5\vec{y} = (b_1, \ldots, b_5)$ has a unique solution

$$\vec{y} = \begin{pmatrix} 4 \cdot 5 \\ 2 \cdot 5 \\ 5 \\ 1 \\ 1 \end{pmatrix}.$$

However $\sum_{i=1,\ldots,5} y_i r_{l_i} = 2 \cdot 5 + 5 + 1 \neq 4 \cdot 5 + 5 + \alpha = b_6^{1,\alpha}$, for any $\alpha \in \{0, 1, 2\}$. Thus X does not span $\vec{b}^{1,0}$, $\vec{b}^{1,1}$, $\vec{b}^{1,2}$. Similarly X does not span $\vec{b}^{2,0}$, $\vec{b}^{2,1}$, $\vec{b}^{2,2}$.

References

1. Boros, E., Elbassioni, K., Gurvich, V., Khachiyan, L., Makino, K.: Generating paths and cuts in multi-pole (di)graphs. In: Fiala, J., Koubek, V., Kratochvil, J. (eds.) Mathematical Foundations of Computer Science MFCS, Prague, Czech Republic, August 22–27, 2004. Lecture Notes in Computer Science, vol. 3153, pp. 298–309. Springer, Berlin (2004)
2. Boros, E., Elbassioni, K., Gurvich, V., Khachiyan, L., Makino, K.: On the complexity of some enumeration problems for matroids. SIAM J. Discrete Math. **19**(4), 966–984 (2005)
3. Boros, E., Elbassioni, K., Gurvich, V.: Transversal hypergraphs to perfect matchings in bipartite graphs: characterization and generation algorithms. J. Graph Theory **53**(3), 209–232 (2006)
4. Dahlhaus, E., Johnson, D.S., Papadimitriou, C.H., Seymour, P.D., Yannakakis, M.: The complexity of multiway cuts. In: Proceedings of the 24th ACM Symposium on Theory of Computing, pp. 241–251 (1992)
5. Eiter, T., Gottlob, G.: Identifying the minimal transversals of a hypergraph and related problems. SIAM J. Comput. **24**, 1278–1304 (1995)
6. Fredman, M., Khachiyan, L.: On the complexity of dualization of monotone disjunctive normal forms. J. Algorithms **21**, 618–628 (1996)
7. Hu, T.C.: Multicomodity network flows. Oper. Res. **11**, 344–360 (1963)
8. Johnson, D.S., Papadimitriou, C.H.: On generating all maximal independent sets. Inf. Process. Lett. **27**, 119–123 (1988)
9. Lawler, E., Lenstra, J.K., Rinnooy Kan, A.H.G.: Generating all maximal independent sets NP-hardness and polynomial-time algorithms. SIAM J. Comput. **9**, 558–565 (1980)
10. Oxley, J.G.: Matroid Theory. Oxford University Press, Oxford (1992)
11. Read, R.C., Tarjan, R.E.: Bounds on backtrack algorithms for listing cycles, paths, and spanning trees. Networks **5**, 237–252 (1975)
12. Schrijver, A.: Combinatorial Optimization Polyhedra and Efficiency, vol. B. Springer, Berlin (2003). p. 654
13. Schwikowski, B., Speckenmeyer, E.: On enumerating all minimal solutions of feedback problems. Discrete Appl. Math. **117**(1–3), 253–265 (2002)
14. Shioura, A., Tamura, A.: Efficiently scanning all spanning trees of an undirected graph. J. Oper. Res. **38**, 331–344 (1995)
15. Tarjan, R.: A note on finding the bridges of a graph. Inf. Process. Lett. **2**, 160–161 (1974)
16. Tsukiyama, S., Shirakawa, I., Ozaki, H., Ariyoshi, H.: An algorithm to enumerate all cutsets of a graph in linear time per cutset. J. Assoc. Comput. Mach. **27**, 619–632 (1980)
17. Vazirani, V.: Approximation Algorithms. Springer, Berlin (2001)
18. Welsh, D.J.A.: Matroid Theory. Academic, London (1976)